```
VVV         VVV  MMM         MMM     SSSSSSSSSSSS  LLL                 IIIIIIIII     BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSSSSS  LLL                 IIIIIIIII     BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSSSSS  LLL                 IIIIIIIII     BBBBBBBBBBBB
VVV         VVV  MMMMMM   MMMMMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMMMMM   MMMMMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMMMMM   MMMMMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM   MMM   MMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM   MMM   MMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM   MMM   MMM  SSS              LLL                    III        BBB         BBB
VVV         VVV  MMM         MMM     SSSSSSSSS     LLL                    III        BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSS     LLL                    III        BBBBBBBBBBBB
VVV         VVV  MMM         MMM     SSSSSSSSS     LLL                    III        BBBBBBBBBBBB
VVV         VVV  MMM         MMM              SSS  LLL                    III        BBB         BBB
VVV         VVV  MMM         MMM              SSS  LLL                    III        BBB         BBB
VVV         VVV  MMM         MMM              SSS  LLL                    III        BBB         BBB
   VVV   VVV     MMM         MMM              SSS  LLL                    III        BBB         BBB
   VVV   VVV     MMM         MMM              SSS  LLL                    III        BBB         BBB
   VVV   VVV     MMM         MMM              SSS  LLL                    III        BBB         BBB
      VVV        MMM         MMM  SSSSSSSSSSSS     LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB
      VVV        MMM         MMM  SSSSSSSSSSSS     LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB
      VVV        MMM         MMM  SSSSSSSSSSSS     LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB
```

```
**FILE**ID**OBJFMT                                                    N 15




  OOOOOO    BBBBBBBB          JJ  FFFFFFFFFF  MM      MM  TTTTTTTTTT
  OOOOOO    BBBBBBBB          JJ  FFFFFFFFFF  MM      MM  TTTTTTTTTT
OO      OO  BB      BB        JJ  FF          MMMM  MMMM      TT
OO      OO  BB      BB        JJ  FF          MMMM  MMMM      TT
OO      OO  BB      BB        JJ  FF          MM  MM  MM      TT
OO      OO  BB      BB        JJ  FF          MM  MM  MM      TT
OO      OO  BBBBBBBB          JJ  FFFFFFFF    MM      MM      TT
OO      OO  BBBBBBBB          JJ  FFFFFFFF    MM      MM      TT
OO      OO  BB      BB  JJ    JJ  FF          MM      MM      TT
OO      OO  BB      BB  JJ    JJ  FF          MM      MM      TT
OO      OO  BB      BB  JJ    JJ  FF          MM      MM      TT            ....
OO      OO  BB      BB  JJ    JJ  FF          MM      MM      TT            ....
  OOOOOO    BBBBBBBB      JJJJJJ  FF          MM      MM      TT            ....
  OOOOOO    BBBBBBBB      JJJJJJ  FF          MM      MM      TT            ....


  SSSSSSSS  DDDDDDDD    LL
  SSSSSSSS  DDDDDDDD    LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
  SSSSSS    DD      DD  LL
  SSSSSS    DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

```
{
{ Version:       'V04-000'
{
{****************************************************************************
{*                                                                          *
{*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
{*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
{*  ALL RIGHTS RESERVED.                                                    *
{*                                                                          *
{*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
{*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
{*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
{*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
{*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
{*  TRANSFERRED.                                                            *
{*                                                                          *
{*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
{*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
{*  CORPORATION.                                                            *
{*                                                                          *
{*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
{*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
{*                                                                          *
{*                                                                          *
{****************************************************************************

{ MODIFIED BY:
{
{       V03-005 JWT0113         Jim Teague              27-Apr-1983
{               Another new type for the Linker Options Record,
{               LNK$C_SHA, for individually specified shr imgs.
{
{       V03-004 JWT0102         Jim Teague              16-Mar-1983
{               Add a new type to the Linker Options Record, LNK$C_OBJ.
{
{       V03-003 JWT0082         Jim Teague              20-Dec-1982
{               Add V_NESTED to environment flags to clear up the
{               ambiguity of parent environment zero.
{
{       V03-002 ACG0303         Andrew C. Goldstein,    9-Dec-1982  16:02
{               Add FILL attribute to extraneous field names
{
{       V03-001         JWT0037         Jim Teague              18-Jun-1982
{               Add spec for linker options record (LNK)
{
{       V02-008         BLS0096         Benn Schreiber          31-Oct-1981
{               Add shareable image psect type SGPS
{
{       V02-007         BLS0094         Benn Schreiber          31-Oct-1981
{               Add STA_LEPM
{
{       V02-006         BLS0084         Benn Schreiber          21-Sep-1981
{               Make IDC IDMATCH 2 bits, add ERRSEV
{
{       V02-005         BLS0062         Benn Schreiber          28-Jul-1981
{               Correct local symbol definition
```

```
{
{       V02-004         BLS0045         Benn Schreiber          14-Mar-1981
{               Correct store repeated limit to be longword
{
{       V02-003         BLS0037         Benn Schreiber          29-Jan-1981
{               Add rest of new object language commands: local symbols,
{               end of module word psect.
{
{       V02-002         BLS0033         Benn Schreiber          5-Jan-1981
{               Add new definitions for more psects, add literal operators,
{               and ident check.
{
{       V02-001         BLS0011         Benn Schreiber          1-Sep-1980
{               Implement TIR$C_CTL_STKDL to stack debug location.
{---
{
{ Definition file for the VAX/VMS object language
{
module $OBJRECDEF;


aggregate OBJRECDEF structure prefix OBJ$;
    RECTYP byte unsigned;                                   /*First byte always record type
                                                            /*Permissable record types
    constant HDR        equals 0  prefix OBJ tag $C;        /*Module header record
    constant HDR_MHD    equals 0  prefix OBJ tag $C;        /* Main header record
    constant HDR_LNM    equals 1  prefix OBJ tag $C;        /* Language processor record
    constant HDR_SRC    equals 2  prefix OBJ tag $C;        /* Source files description
    constant HDR_TTL    equals 3  prefix OBJ tag $C;        /* Title text
    constant HDR_CPR    equals 4  prefix OBJ tag $C;        /* Copyright text
    constant HDR_MTC    equals 5  prefix OBJ tag $C;        /* Maintenance text
    constant HDR_GTX    equals 6  prefix OBJ tag $C;        /* General text
    constant GSD        equals 1  prefix OBJ tag $C;        /*Global symbol definition record
    constant GSD_PSC    equals 0  prefix OBJ tag $C;        /* P-sect definition
    constant GSD_SYM    equals 1  prefix OBJ tag $C;        /* Symbol (simple) definition
    constant GSD_EPM    equals 2  prefix OBJ tag $C;        /* Entry point definition
    constant GSD_PRO    equals 3  prefix OBJ tag $C;        /* Procedure definition
    constant GSD_SYMW   equals 4  prefix OBJ tag $C;        /* Symbol definition with word psect
    constant GSD_EPMW   equals 5  prefix OBJ tag $C;        /* Entry point definition with word psect
    constant GSD_PROW   equals 6  prefix OBJ tag $C;        /* Procedure definition with word psect
    constant GSD_IDC    equals 7  prefix OBJ tag $C;        /* Random entity check
    constant GSD_ENV    equals 8  prefix OBJ tag $C;        /* Environment definition
    constant GSD_LSY    equals 9  prefix OBJ tag $C;        /* Local symbol definition/reference
    constant GSD_LEPM   equals 10  prefix OBJ tag $C;       /* Local symbol entry point def.
    constant GSD_LPRO   equals 11  prefix OBJ tag $C;       /* Local symbol procedure def.
    constant GSD_SPSC   equals 12  prefix OBJ tag $C;       /* Shareable image psect definition
    constant TIR        equals 2  prefix OBJ tag $C;        /*Text information record
    constant EOM        equals 3  prefix OBJ tag $C;        /*End of module record
    constant DBG        equals 4  prefix OBJ tag $C;        /*Debugger information record
    constant TBT        equals 5  prefix OBJ tag $C;        /*Traceback information record
    constant LNK        equals 6  prefix OBJ tag $C;        /*Linker options record
    constant EOMW       equals 7  prefix OBJ tag $C;        /*End of module record with word psect
    constant MAXRECTYP  equals 7  prefix OBJ tag $C;        /*Last assigned record type
    constant SUBTYP equals . prefix OBJ$ tag K;
    constant SUBTYP equals . prefix OBJ$ tag C;
    SUBTYP byte unsigned;                                   /*Record sub-type byte
    MHD_STRLV byte unsigned;                                    /*Structure level
    MHD_RECSZ_OVERLAY union fill;
        MHD_RECSZ word unsigned;                                /*Maximum record size
        MHD_RECSZ_FIELDS structure fill;
            FILL_T byte dimension 2 fill prefix OBJRECDEF tag $$;
            MHD_NAME character length 0 tag T;                  /*Module name field
                                                                /*Misc. constants
            constant MAXRECSIZ  equals 2048  prefix OBJ tag $C;/*Maximum legal record size
            constant STRLVL     equals 0  prefix OBJ tag $C;/*Structure level
            constant SYMSIZ     equals 31  prefix OBJ tag $C;/*Maximum symbol length
            constant STOREPLIM  equals -1  prefix OBJ tag $C;/*Maximum repeat count on store commands
            constant PSCALILIM  equals 9  prefix OBJ tag $C;/*Maximum p-sect alignment
        end MHD_RECSZ_FIELDS;
    end MHD_RECSZ_OVERLAY;
end OBJRECDEF;

end_module $OBJRECDEF;

module $MHDEF;

/*
/* Module header record (MHD)
/*


aggregate MHDEF structure prefix MHD$;
    RECTYP byte unsigned;                                       /*Record type (OBJ$C_MHD)
    HDRTYP byte unsigned;                                       /*Type field for MHD
                                                                /*Types of header records
    constant MHD        equals 0  prefix MHD tag $C;            /*Main header record
    constant LNM        equals 1  prefix MHD tag $C;            /*Language name and version
    constant SRC        equals 2  prefix MHD tag $C;            /*Source file specification
    constant TTL        equals 3  prefix MHD tag $C;            /*Title text of module
    constant CPR        equals 4  prefix MHD tag $C;            /*Copyright notice
    constant MTC        equals 5  prefix MHD tag $C;            /*Maintenence status
    constant GTX        equals 6  prefix MHD tag $C;            /*General text
    constant MAXHDRTYP  equals 6  prefix MHD tag $C;            /*Maximum allowable type
    STRLVL byte unsigned;                                       /*Structure level
    RECSIZ word unsigned;                                       /*Maximum record size
    NAMLNG byte unsigned;                                       /*Module name length
    NAME character length 31;                                   /*Module name
/*                                                   /*Module version (ASCIC)
/*                                                   /*Creation date/time (17 bytes)
/*                                                   /*Time of last patch (17 bytes)
end MHDEF;

end_module $MHDEF;

module $EOMDEF;

/*
/* End of module record (EOM)
/*


aggregate EOMDEF structure prefix EOM$;
```

```
    RECTYP byte unsigned;                                       /*Record type (OBJ$C_EOM)
    COMCOD byte unsigned;                                       /*Compiler completion code
                                                                /*Values
    constant SUCCESS    equals 0  prefix EOM tag $C;            /*Successful (no errors)
    constant WARNING    equals 1  prefix EOM tag $C;            /*Warnings issued
    constant ERROR      equals 2  prefix EOM tag $C;            /*Errors detected
    constant ABORT      equals 3  prefix EOM tag $C;            /*Abort the link
    constant EOMMIN equals . prefix EOM$ tag K;                 /*Min length of EOM record
    constant EOMMIN equals . prefix EOM$ tag C;                 /*Min length of EOM record
    PSINDX byte unsigned;                                       /*P-sect of transfer address
    TFRADR longword unsigned;                                   /*Transfer address
    constant EOMMX1 equals . prefix EOM$ tag K;                 /*Length of EOM record w/o transfer flags
    constant EOMMX1 equals . prefix EOM$ tag C;                 /*Length of EOM record w/o transfer flags
    TFRFLG_OVERLAY union fill;
        TFRFLG byte unsigned;                                   /*Transfer address flags
        constant EOMMAX equals . prefix EOM$ tag K;             /*Maximum length of EOM record
        constant EOMMAX equals . prefix EOM$ tag C;             /*Maximum length of EOM record
        TFRFLG_BITS structure fill;
            WKTFR bitfield mask;                                /*Transfer address is weak
        end TFRFLG_BITS;
    end TFRFLG_OVERLAY;
end EOMDEF;

end_module $EOMDEF;

module $EOMWDEF;
/*
/* End of module record with word of psect (EOMW)
/*


aggregate EOMWDEF structure prefix EOMW$;
    RECTYP byte unsigned;                                       /*Record type (OBJ$C_EOM)
    COMCOD byte unsigned;                                       /*Compiler completion code
    constant EOMMIN equals . prefix EOMW$ tag K;                /*Min length of EOM record
    constant EOMMIN equals . prefix EOMW$ tag C;                /*Min length of EOM record
    PSINDX word unsigned;                                       /*P-sect of transfer address
    TFRADR longword unsigned;                                   /*Transfer address
    constant EOMMX1 equals . prefix EOMW$ tag K;                /*Length of EOMW record w/o transfer flags
    constant EOMMX1 equals . prefix EOMW$ tag C;                /*Length of EOMW record w/o transfer flags
    TFRFLG_OVERLAY union fill;
        TFRFLG byte unsigned;                                   /*Transfer address flags
        constant EOMMAX equals . prefix EOMW$ tag K;            /*Maximum length of EOMW record
        constant EOMMAX equals . prefix EOMW$ tag C;            /*Maximum length of EOMW record
        TFRFLG_BITS structure fill;
            WKTFR bitfield mask;                                /*Transfer address is weak
        end TFRFLG_BITS;
    end TFRFLG_OVERLAY;
end EOMWDEF;

end_module $EOMWDEF;

module $LNKDEF;

/*
/* Linker Options Record (LNK)
```

```
/*


aggregate LNKDEF structure prefix LNK$;
    RECTYP byte unsigned;                                    /* record type LNK
    LNKTYP byte unsigned;                                    /* sub record type
    constant OLB        equals 0  prefix LNK tag $C;         /* object library spec
    constant SHR        equals 1  prefix LNK tag $C;         /* shareable image library spec
    constant OLI        equals 2  prefix LNK tag $C;         /* object library with inclusion list
    constant OBJ        equals 3  prefix LNK tag $C;         /* object file or symbol table file
    constant SHA        equals 4  prefix LNK tag $C;         /* individually specified shr img
    constant MAXRECTYP  equals 4  prefix LNK tag $C;         /* highest current record type
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;
        FLAGS_BITS structure fill;
            SELSER bitfield mask;                            /* selectively searched (LNK$C_OBJ)
            LIBSRCH bitfield mask;
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    NAMLNG_OVERLAY union fill;
        NAMLNG word unsigned;                                /* length of filespec name
        NAMLNG_FIELDS structure fill;
            FILL_1 byte dimension 2 fill prefix LNKDEF tag $$;
            NAME character length 0 tag T;                   /* actual name
        end NAMLNG_FIELDS;
    end NAMLNG_OVERLAY;
end LNKDEF;

end_module $LNKDEF;

module $GSDEF;

/*
/* Global symbol definition record (GSD)
/*


aggregate GSDEF structure prefix GSD$;
    RECTYP byte unsigned;                                    /*Record type (OBJ$C_GSD)
    constant ENTRIES equals . prefix GSD$ tag K;             /*Offset to first entry in record
    constant ENTRIES equals . prefix GSD$ tag C;             /*Offset to first entry in record
    GSDTYP byte unsigned;                                    /*Type of entry (first byte of entry)
    constant PSC        equals 0  prefix GSD tag $C;         /*Psect definition
    constant SYM        equals 1  prefix GSD tag $C;         /*Symbol specification
    constant EPM        equals 2  prefix GSD tag $C;         /*Entry point and mask definition
    constant PRO        equals 3  prefix GSD tag $C;         /*Procedure with formal arguments
    constant SYMW       equals 4  prefix GSD tag $C;         /*Symbol specification with word psect
    constant EPMW       equals 5  prefix GSD tag $C;         /*Entry point mask with word psect
    constant PROW       equals 6  prefix GSD tag $C;         /*Procedure with word psect
    constant IDC        equals 7  prefix GSD tag $C;         /*Random entity check
    constant ENV        equals 8  prefix GSD tag $C;         /*Define environment
    constant LSY        equals 9  prefix GSD tag $C;         /*Local symbol
    constant LEPM       equals 10  prefix GSD tag $C;        /*Local symbol entry point definition
    constant LPRO       equals 11  prefix GSD tag $C;        /*Local symbol procedure definition
    constant SPSC       equals 12  prefix GSD tag $C;   /*Shareable image psect definition
    constant MAXRECTYP  equals 12  prefix GSD tag $C;   /*Maximum entry type defined
end GSDEF;

end_module $GSDEF;

module $GPSDEF;

/*
/* GSD entry - P-section definition
/*


aggregate GPSDEF structure prefix GPS$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Typ field
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix GPSDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    ALIGN byte unsigned;                                /*P-sect alignment
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*P-sect flags
        FLAGS_BITS structure fill;
            PIC bitfield mask;                          /*Position independent
            LIB bitfield mask;                          /*From a shareable image
            OVR bitfield mask;                          /*Overlaid memory allocation
            REL bitfield mask;                          /*Relocatable
            GBL bitfield mask;                          /*Global scope
            SHR bitfield mask;                          /*Shareable
            EXE bitfield mask;                          /*Executable
            RD bitfield mask;                           /*Readable
            WRT bitfield mask;                          /*Writeable
            VEC bitfield mask;                          /*Vector psect
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    ALLOC longword unsigned;                            /*Length of this contribution
    NAMLNG byte unsigned;                               /*Length of p-sect name
    constant NAME equals . prefix GPS$ tag K;
    constant NAME equals . prefix GPS$ tag C;
    NAME character length 31;                           /*Name field
end GPSDEF;

end_module $GPSDEF;

module $SGPSDEF;
/*
/* GSD entry - P-section definition in shareable image
/*


aggregate SGPSDEF structure prefix SGPS$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Typ field
        GSDTYP_FIELDS structure fill;
```

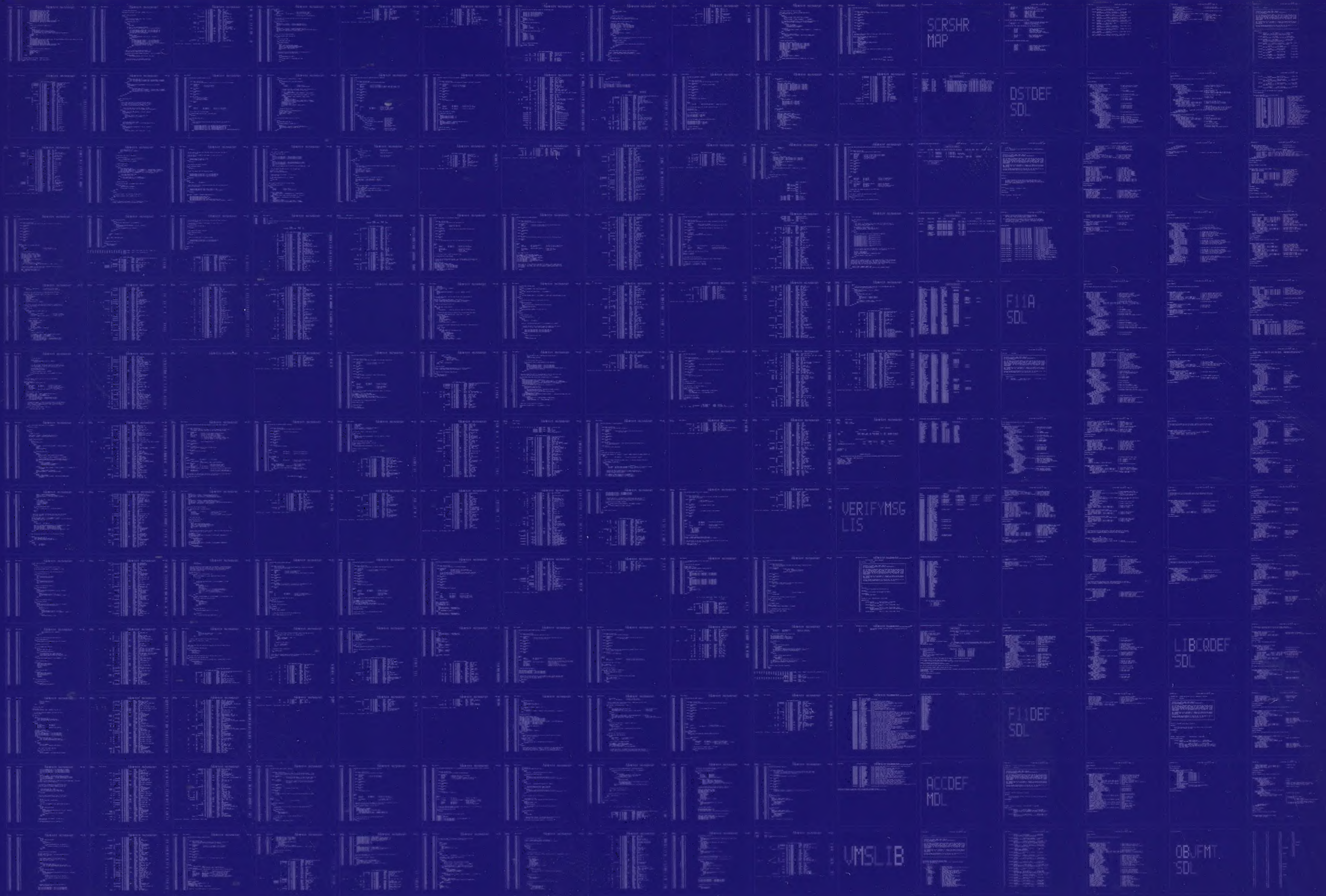
```
            START character length 0 tag T;
            FILL_1 byte fill prefix SGPSDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    ALIGN byte unsigned;                                    /*P-sect alignment
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                                /*P-sect flags
        FLAGS_BITS structure fill;
            PIC bitfield mask;                              /*Position independent
            LIB bitfield mask;                              /*From a shareable image
            OVR bitfield mask;                              /*Overlaid memory allocation
            REL bitfield mask;                              /*Relocatable
            GBL bitfield mask;                              /*Global scope
            SHR bitfield mask;                              /*Shareable
            EXE bitfield mask;                              /*Executable
            RD bitfield mask;                               /*Readable
            WRT bitfield mask;                              /*Writeable
            VEC bitfield mask;                              /*Vector psect
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    ALLOC longword unsigned;                                /*Length of this psect in shr image
    BASE longword unsigned;                                 /*Base of this psect in shr image
    NAMLNG byte unsigned;                                   /*Length of p-sect name
    constant NAME equals . prefix SGPS$ tag K;
    constant NAME equals . prefix SGPS$ tag C;
    NAME character length 31;                               /*Name field
end SGPSDEF;

end_module $SGPSDEF;

module $GSYDEF;
/*
/* GSD entry - Symbol definition
/*
/* common to definitions, references, and entry
/* point definitions.
/*



aggregate GSYDEF structure prefix GSY$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                               /*Type field
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix GSYDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                    /*Symbol data type
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                                /*Symbol flags
        FLAGS_BITS structure fill;
            WEAK bitfield mask;                             /*Weak symbol
            DEF bitfield mask;                              /*Definition
            UNI bitfield mask;                              /*Universal
            REL bitfield mask;                              /*Relocatable
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
end GSYDEF;

end_module $GSYDEF;

module $SRFDEF;
/*
/* Symbol reference (SYM$M_DEF in GSY$W_FLAGS is 0)
/*


aggregate SRFDEF structure prefix SRF$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                              /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix SRFDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                   /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                   /*Maps over GSY$W_FLAGS
    NAMLNG byte unsigned;                                  /*Length of symbol name
    constant NAME equals . prefix SRF$ tag K;
    constant NAME equals . prefix SRF$ tag C;
    NAME character length 31;                              /*Symbol name
end SRFDEF;

end_module $SRFDEF;

module $SDFDEF;
/*
/* Symbol definition
/*


aggregate SDFDEF structure prefix SDF$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                              /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix SDFDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                   /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                   /*Maps over GSY$W_FLAGS
    PSINDX byte unsigned;                                  /*Owning psect number
    "VALUE" longword unsigned;                             /*Value of symbol
    NAMLNG byte unsigned;                                  /*Length of name
    constant NAME equals . prefix SDF$ tag K;
    constant NAME equals . prefix SDF$ tag C;
    NAME character length 31;                              /*Symbol name
end SDFDEF;

end_module $SDFDEF;
```

```
module $EPMDEF;
/*
/* GSD entry - Entry point definition
/*


aggregate EPMDEF structure prefix EPM$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix EPMDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over GSY$W_FLAGS
    PSINDX byte unsigned;                               /*Maps over SDF$B_PSINDX
    ADDRS longword unsigned;                            /*Entry point address, maps over SDF$L_VALUE
    "MASK" word unsigned;                               /*Entry point mask
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix EPM$ tag K;
    constant NAME equals . prefix EPM$ tag C;
    NAME character length 31;                           /*Symbol name
end EPMDEF;

end_module $EPMDEF;

module $PRODEF;
/*
/* GSD entry - Procedure definition
/*


aggregate PRODEF structure prefix PRO$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix PRODEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over GSY$W_FLAG
    PSINDX byte unsigned;                               /*Maps over SDF$B_PS
    ADDRS longword unsigned;                            /*Entry point address   aps over SDF$L_VALUE
    "MASK" word unsigned;                               /*Entry point mask
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix PRO$ tag K;
    constant NAME equals . prefix PRO$ tag C;
    NAME character length 31;                           /*Symbol name
end PRODEF;

end_module $PRODEF;

module $FMLDEF;
/*
```

```
/* Appended to a procedure definition are the formal arguments:
/*          FML$ - The fixed part of the formal arguments description
/*


aggregate FMLDEF structure prefix FML$;
    MINARGS byte unsigned;                              /*Minimum number of arguments
    MAXARGS byte unsigned;                              /*Maximum which include function if procedure is one
    constant SIZE equals . prefix FML$ tag K;
    constant SIZE equals . prefix FML$ tag C;
end FMLDEF;

end_module $FMLDEF;

module $ARGDEF;
/*
/*          ARG$ - The argument descriptors
/*


aggregate ARGDEF structure prefix ARG$;
    VALCTL_OVERLAY union fill;
        VALCTL byte unsigned;                           /*Validation control byte
        VALCTL_BITS structure fill;
            PASSMECH bitfield length 2;                 /*Passing mechanism
        end VALCTL_BITS;
                                                        /* Passing mechanisms
        constant UNKNOWN        equals 0  prefix ARG tag $C;/* Unspecified or unknown
        constant "VALUE"        equals 1  prefix ARG tag $C;/* Passed by value
        constant "REF"  equals 2  prefix ARG tag $C;    /* Passed by reference
        constant DESC   equals 3  prefix ARG tag $C;    /* Passed by descriptor
    end VALCTL_OVERLAY;
    BYTECNT byte unsigned;                              /*Remaining byte count
    constant SIZE equals . prefix ARG$ tag K;
    constant SIZE equals . prefix ARG$ tag C;
end ARGDEF;

end_module $ARGDEF;

module $SDFWDEF;

/*
/* Symbol definition with word of psect value
/*


aggregate SDFWDEF structure prefix SDFW$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix SDFWDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over GSY$W_FLAGS
    PSINDX word unsigned;                               /*Owning psect number
    "VALUE" longword unsigned;                          /*Value of symbol
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix SDFW$ tag K;
    constant NAME equals . prefix SDFW$ tag C;
    NAME character length 31;                           /*Symbol name
end SDFWDEF;

end_module $SDFWDEF;

module $EPMWDEF;
/*
/* GSD entry - Entry point definition with word of psect value
/*


aggregate EPMWDEF structure prefix EPMW$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix EPMWDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over GSY$W_FLAGS
    PSINDX word unsigned;                               /*Maps over SDFW$W_PSINDX
    ADDRS longword unsigned;                            /*Entry point address, maps over SDFW$L_VALUE
    "MASK" word unsigned;                               /*Entry point mask
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix EPMW$ tag K;
    constant NAME equals . prefix EPMW$ tag C;
    NAME character length 31;                           /*Symbol name
end EPMWDEF;

end_module $EPMWDEF;

module $PROWDEF;
/*
/* GSD entry - Procedure definition with word of psect value
/*


aggregate PROWDEF structure prefix PROW$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over GSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix PROWDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Maps over GSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over GSY$W_FLAGS
    PSINDX word unsigned;                               /*Maps over SDFW$W_PSINDX
    ADDRS longword unsigned;                            /*Entry point address, maps over SDFW$L_VALUE
    "MASK" word unsigned;                               /*Entry point mask
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix PROW$ tag K;
    constant NAME equals . prefix PROW$ tag C;
    NAME character length 31;                           /*Symbol name
end PROWDEF;

end_module $PROWDEF;

module $IDCDEF;
/*
/* IDC - Random entity ident consistency check
/*


aggregate IDCDEF structure prefix IDC$;
    GSDTYP byte unsigned;                               /*Type field
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*Flags
        FLAGS_BITS structure fill;
            BINIDENT bitfield;                          /*Ident is binary longword rather than ASCIC
            IDMATCH bitfield length 2;                  /*Field for ident match control if binary ident
            ERRSEV bitfield length 3;                   /*Error severity (default is warning-0)
        end FLAGS_BITS;
                                                        /*Match control values
        constant(
              LEQ
              EQUAL
            ) equals 0 increment 1 prefix IDC tag $C;
    end FLAGS_OVERLAY;
    NAMLNG_OVERLAY union fill;
        NAMLNG byte unsigned;                           /*Length of entity name
        NAMLNG_FIELDS structure fill;
            FILL_1 byte fill prefix IDCDEF tag $$;
            NAME character length 0 tag T;              /*
                                                        /* Followed by entity name
                                                        /* Followed by
                                                        /*      byte of ident length
                                                        /*              ident string (length = string length)
                                                        /*                      or
                                                        /*              ident binary value (length = 4)
                                                        /* Followed by byte of length of name of object
                                                        /* Followed by the object name
        end NAMLNG_FIELDS;
    end NAMLNG_OVERLAY;
end IDCDEF;

end_module $IDCDEF;

module $ENVDEF;
/*
/* ENV - Define/reference an environment
/*


aggregate ENVDEF structure prefix ENV$;
    GSDTYP byte unsigned;                               /*Type field
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*Environment flags
        FLAGS_BITS structure fill;
            DEF bitfield mask;                          /*Definition of environment
            NESTED bitfield mask;                       /*Nested environment if set
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    ENVINDX word unsigned;                              /*Index of parent environment
    NAMLNG byte unsigned;                               /*Length of environment name
    NAME character length 31;                           /*Environment name
end ENVDEF;

end_module $ENVDEF;

module $LSYDEF;
/*
/* LSY - Module-Local symbol definition
/*
/* Common to definitions, references, entry points, and procedure definitions
/*


aggregate LSYDEF structure prefix LSY$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Type field
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix LSYDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Symbol type
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*Symbol flags
        FLAGS_BITS structure fill;
            WEAK bitfield mask;                         /*Weak symbol (not used)
            DEF bitfield mask;                          /*Defined symbol
            UNI bitfield mask;                          /*Universal (not used)
            REL bitfield mask;                          /*Relocatable
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    ENVINDX word unsigned;                              /*Environment index
end LSYDEF;

end_module $LSYDEF;

module $LSRFDEF;
/*
/* Module-local Symbol reference (LSY$M_DEF in LSY$W_FLAGS is 0)
/*


aggregate LSRFDEF structure prefix LSRF$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over LSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix LSRFDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                        /*Maps over LSY$B_DATYP
    FLAGS word unsigned;                                        /*Maps over LSY$W_FLAGS
    ENVINDX word unsigned;                                      /*Maps over LSY$W_ENVINDX
    NAMLNG byte unsigned;                                       /*Length of symbol name
    constant NAME equals . prefix LSRF$ tag K;
    constant NAME equals . prefix LSRF$ tag C;
    NAME character length 31;                                   /*Symbol name
end LSRFDEF;

end_module $LSRFDEF;

module $LSDFDEF;
/*
/* Module-local Symbol definition
/*


aggregate LSDFDEF structure prefix LSDF$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                                   /*Maps over LSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix LSDFDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                        /*Maps over LSY$B_DATYP
    FLAGS word unsigned;                                        /*Maps over LSY$W_FLAGS
    ENVINDX word unsigned;                                      /*Environment index symbol defined in
    PSINDX word unsigned;                                       /*Owning psect number
    "VALUE" longword unsigned;                                  /*Value of symbol
    NAMLNG byte unsigned;                                       /*Length of name
    constant NAME equals . prefix LSDF$ tag K;
    constant NAME equals . prefix LSDF$ tag C;
    NAME character length 31;                                   /*Symbol name
end LSDFDEF;

end_module $LSDFDEF;

module $LEPMDEF;
/*
/* GSD entry - Module local entry point definition
/*


aggregate LEPMDEF structure prefix LEPM$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                                   /*Maps over LSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix LEPMDEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                        /*Maps over LSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over LSY$W_FLAGS
    ENVINDX word unsigned;                              /*Environment index symbol defined in
    PSINDX word unsigned;                               /*Maps over LSDF$W_PSINDX
    ADDRS longword unsigned;                            /*Entry point address, maps
                                                        /* over LSDF$L_VALUE
    'MASK' word unsigned;                               /*Entry point mask
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix LEPM$ tag K;
    constant NAME equals . prefix LEPM$ tag C;
    NAME character length 31;                           /*Symbol name
end LEPMDEF;

end_module $LEPMDEF;

module $LPRODEF;
/*
/* GSD entry - Module Local Procedure definition
/*


aggregate LPRODEF structure prefix LPRO$ origin FILL_1;
    GSDTYP_OVERLAY union fill;
        GSDTYP byte unsigned;                           /*Maps over LSY$B_GSDTYP
        GSDTYP_FIELDS structure fill;
            START character length 0 tag T;
            FILL_1 byte fill prefix LPRODEF tag $$;
        end GSDTYP_FIELDS;
    end GSDTYP_OVERLAY;
    DATYP byte unsigned;                                /*Maps over LSY$B_DATYP
    FLAGS word unsigned;                                /*Maps over LSY$W_FLAGS
    ENVINDX word unsigned;                              /*Environment index symbol defined in
    PSINDX word unsigned;                               /*Maps over LSDF$W_PSINDX
    ADDRS longword unsigned;                            /*Entry point address, maps
                                                        /* over LSDF$L_VALUE
    'MASK' word unsigned;                               /*Entry point mask
    NAMLNG byte unsigned;                               /*Length of name
    constant NAME equals . prefix LPRO$ tag K;
    constant NAME equals . prefix LPRO$ tag C;
    NAME character length 31;                           /*Symbol name
end LPRODEF;

end_module $LPRODEF;

module $TIRDEF;
/*
/* Text, information and relocation record (TIR)
/*



aggregate TIRDEF union prefix TIR$;
    RECTYP byte unsigned;                               /*Record type (OBJ$C_TIR)

                                                        /* Define relocation commands

    constant STA_GBL    equals 0  prefix TIR tag $C;    /*Stack global symbol value
    constant STA_SB      equals 1  prefix TIR tag $C;    /*Stack signed byte
    constant STA_SW      equals 2  prefix TIR tag $C;    /*Stack signed word
    constant STA_LW      equals 3  prefix TIR tag $C;    /*Stack longword
    constant STA_PB      equals 4  prefix TIR tag $C;    /*Stack psect base plus byte offset
    constant STA_PW      equals 5  prefix TIR tag $C;    /*Stack psect base plus word offset
    constant STA_PL      equals 6  prefix TIR tag $C;    /*Stack psect base plus longword offset
    constant STA_UB      equals 7  prefix TIR tag $C;    /*Stack unsigned byte
    constant STA_UW      equals 8  prefix TIR tag $C;    /*Stack unsigned word
    constant STA_BFI     equals 9  prefix TIR tag $C;    /*Stack byte from image
    constant STA_WFI     equals 10  prefix TIR tag $C;   /*Stack word from image
    constant STA_LFI     equals 11  prefix TIR tag $C;   /*Stack longword from image
    constant STA_EPM     equals 12  prefix TIR tag $C;   /*Stack entry point mask
    constant STA_CKARG   equals 13  prefix TIR tag $C;   /*Stack result of argument checking (true or false)
    constant STA_WPB     equals 14  prefix TIR tag $C;   /*Stack psect base plus byte offset -- word psect number
    constant STA_WPW     equals 15  prefix TIR tag $C;   /*Stack psect base plus word offset -- word psect number
    constant STA_WPL     equals 16  prefix TIR tag $C;   /*Stack psect base plus longword offset -- word of psect number
    constant STA_LSY     equals 17  prefix TIR tag $C;   /*Stack local symbol value
    constant STA_LIT     equals 18  prefix TIR tag $C;   /*Stack literal
    constant STA_LEPM    equals 19  prefix TIR tag $C;   /*Stack local symbol entry point mask
    constant MAXSTACOD   equals 19  prefix TIR tag $C;   /*Last assigned code of stack group
    constant MINSTOCOD   equals 20  prefix TIR tag $C;   /*First assigned store command code
    constant STO_SB      equals 20  prefix TIR tag $C;   /*Store signed byte
    constant STO_SW      equals 21  prefix TIR tag $C;   /*Store signed word
    constant STO_L       equals 22  prefix TIR tag $C;   /*Store longword
    constant STO_BD      equals 23  prefix TIR tag $C;   /*Store byte displaced
    constant STO_WD      equals 24  prefix TIR tag $C;   /*Store word displaced
    constant STO_LD      equals 25  prefix TIR tag $C;   /*Store longword displaced
    constant STO_LI      equals 26  prefix TIR tag $C;   /*Store short literal
    constant STO_PIDR    equals 27  prefix TIR tag $C;   /*Store pos. indep. data reference
    constant STO_PICR    equals 28  prefix TIR tag $C;   /*Store pos. indep. code reference
    constant STO_RSB     equals 29  prefix TIR tag $C;   /*Store repeated signed byte
    constant STO_RSW     equals 30  prefix TIR tag $C;   /*Store repeated signed word
    constant STO_RL      equals 31  prefix TIR tag $C;   /*Store repeated longword
    constant STO_VPS     equals 32  prefix TIR tag $C;   /*Store arbitrary field
    constant STO_USB     equals 33  prefix TIR tag $C;   /*Store unsigned byte
    constant STO_USW     equals 34  prefix TIR tag $C;   /*Store unsigned word
    constant STO_RUB     equals 35  prefix TIR tag $C;   /*Store repeated unsigned byte
    constant STO_RUW     equals 36  prefix TIR tag $C;   /*Store repeated unsigned word
    constant STO_B       equals 37  prefix TIR tag $C;   /*Store byte
    constant STO_W       equals 38  prefix TIR tag $C;   /*Store word
    constant STO_RB      equals 39  prefix TIR tag $C;   /*Store repeated byte
    constant STO_RW      equals 40  prefix TIR tag $C;   /*Store repeated word
    constant STO_RIVB    equals 41  prefix TIR tag $C;   /*Store repeated immediate variable bytes
    constant STO_PIRR    equals 42  prefix TIR tag $C;   /*Store pos. indep. relative reference
    constant MAXSTOCOD   equals 42  prefix TIR tag $C;   /*Last assigned store command code
    constant MINOPRCOD   equals 50  prefix TIR tag $C;   /*First assigned operator command code
    constant OPR_NOP     equals 50  prefix TIR tag $C;   /*No-op
    constant OPR_ADD     equals 51  prefix TIR tag $C;   /*Add
    constant OPR_SUB     equals 52  prefix TIR tag $C;   /*Subtract
    constant OPR_MUL     equals 53  prefix TIR tag $C;   /*Multiply
    constant OPR_DIV     equals 54  prefix TIR tag $C;   /*Divide
    constant OPR_AND     equals 55  prefix TIR tag $C;   /*Logical AND
    constant OPR_IOR     equals 56  prefix TIR tag $C;   /*Logical inclusive OR
    constant OPR_EOR     equals 57  prefix TIR tag $C;   /*Logical exclusive OR
    constant OPR_NEG     equals 58  prefix TIR tag $C;   /*Negate
    constant OPR_COM     equals 59  prefix TIR tag $C;   /*Complement
    constant OPR_INSV    equals 60  prefix TIR tag $C;   /*Insert bit field
    constant OPR_ASH     equals 61  prefix TIR tag $C;   /*Arithmetic shift
    constant OPR_USH     equals 62  prefix TIR tag $C;   /*Unsigned shift
    constant OPR_ROT     equals 63  prefix TIR tag $C;   /*Rotate
    constant OPR_SEL     equals 64  prefix TIR tag $C;   /*Select one of three longwords on top of stack
    constant OPR_REDEF   equals 65  prefix TIR tag $C;   /*Redefine this symbol after pass 2
    constant OPR_DFLIT   equals 66  prefix TIR tag $C;   /*Define a literal
    constant MAXOPRCOD   equals 66  prefix TIR tag $C;   /*Last assigned operator command code
    constant MINCTLCOD   equals 80  prefix TIR tag $C;   /*First assigned control command code
    constant CTL_SETRB   equals 80  prefix TIR tag $C;   /*Set relocation base
    constant CTL_AUGRB   equals 81  prefix TIR tag $C;   /*Augment relocation base
    constant CTL_DFLOC   equals 82  prefix TIR tag $C;   /*Define debug location
    constant CTL_STLOC   equals 83  prefix TIR tag $C;   /*Set debug location
    constant CTL_STKDL   equals 84  prefix TIR tag $C;   /*Stack debug location
    constant MAXCTLCOD   equals 84  prefix TIR tag $C;   /*Last assigned control command code
end TIRDEF;

end_module $TIRDEF;
```

0432 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



SCRSHR
MAP

DSTDEF
SDL

F11A
SDL

VERIFYMSG
LIS

LIBCQDEF
SDL

F11DEF
SDL

ACCDEF
MDL

VMSLIB

OBJFMT
SDL

0433 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

STARDEFFL
SDL

OPCDEF
SDL

SCRDEF
SDL

OPDEF
SDL

SRMDEF
SDL

STARDEFMP
SDL

STARDEFQZ
SDL

STARDEFAE
SDL